Searching for Optimal Process Routes : A Reinforcement Learning Approach

Ahmad Khan, Alexei Lapkin*

Department of Chemical Engineering and Biotechnology, University of Cambridge , UK

Abstract

Developing optimisation tools is a key target in supporting computer-aided process design as the complexity of the designed space grows beyond conventional unit operations. A process design problem can be formulated as a search of an optimal processing route in the thermodynamic state space, going from feedstock to products. This paper describes a design architecture that enables *reinforcement learning agent* to use trial-and-error to narrow its search to the most promising routes, rather than exhaustively enumerating solutions. In each iteration, the agent employs previously collected data to guide the search for new trajectories. This is successfully demonstrated in a hydrogen production process using both conventional and intensified process design principles. The agent outperformed standard nonlinear optimisation methods in competitive computational time. Limitations and future work are discussed.

Keywords: process design, process intensification, process systems engineering, reinforcement learning, machine learning

1. Introduction

Process design, while being one of the most complex tasks in the development of efficient chemical plants, is also of increasing importance due to the need to transform chemical manufacturing to a more sustainable model. The problem of process design can be approached computationally by searching the design space for the solution that best satisfies certain design goals. [1]. The difficulty lies in the need to balance the size of the considered solution space with the search speed. This is especially significant in the case of more complex problems, such as

*. Corresponding author
Email address: al35@cam.ac.uk (Alexei Lapkin)

process intensification (PI), which move from optimising unit operations into design of new processes via optimising first-principle physical models, like heat transfer, reaction rate and phase separation [2–4]. Rigorous mathematical programming methods are still limited to designing small- albeit novel- intensive processes [3, 5]. Linking these individual processes into optimal integrated flowsheets and plant-level designs, remains a significant challenge.

One sensible approach is to speed up the search by focusing on promising regions in the solution space. While not rigorous, heuristic methods can provide systematic shortcuts and produce viable solutions. Lutze *et. al.* demonstrated a hierarchical methodology to search for intensive designs in a large solution space [4]. Their approach, however, requires extensive human intervention in filtering the options. Several established algorithms involve dividing the solution space e.g. branch-and-bound methods, or dividing the process itself into smaller sequential steps, as illustrated by Chang *et. al.* [6]. These techniques do not exploit the structural features of the space, rendering the search a challenging task [7, 8].

Improving search is one of the applications of artificial intelligence (AI) methods : an algorithm would learn the topography of the search space and employ this information to make decisions, and to search more efficiently [1, 9]. The goal of such learning is to "minimize the total cost of problem solving, trading off computational expense and the [objective function]" [9]. Thus, an algorithm would be able to adapt to more complexity in an optimisation problem. Reinforcement Learning (RL) is a subfield of AI in which an agent aims to learn optimal action policies for different situations. It does so by distilling experience from past interactions with its environment [9, 10]. As such, it learns how to achieve objectives without needing to enumerate all possibilities. An RL agent is similar to an expert system in that they both make decisions based on a knowledge-base or a set of rules. However, expert systems require exact knowledge input from humans, while RL can learn to act optimally from experience [9, 11].

This paper presents an example implementation of the RL paradigm in optimisation-based process design. The design space considered is defined as the set of all possible processing pathways that can transform an arbitrary mass in the feed stream into products. First, we describe the RL methodology and introduce a process environment to test routes, a growing database of visited routes, and a decision-making agent that interacts with the environment based on the available information in the database. We then test this iterative learning cycle

in a hydrogen production process using conventional unit operations and fundamental physical phenomena. Finally, we compare results with random search and with standard mathematical programming packages, and reflect on the work's strengths, weaknesses and future potential.

2. Methodology

2.1. Reinforcement Learning (RL) in Chemical Process Design

The goal for an RL agent is to find an optimal policy that maps states to proper actions, maximising the reward of taking these actions. That is, it learns to behave favourably by interacting with its environment. At the beginning, the agent does not know which actions are appropriate - but as trials accumulate, it forms a better understanding of the environment based on which it can decide, for example, that taking action a_x in state s_n is better than taking action a_y. This is because the latter action was previously found detrimental in a similar state s_m. The more data the agent collects and the better its learning mechanism is, the faster it will converge to an optimal policy.

A chemical process is divided into sequential operating steps that connect feed to products by transforming intermediate streams. This results in a combinatorial optimisation problem, depicted schematically in Figure 1, with the objective of finding the optimal sequence of actions that maximises revenue while minimising action costs.

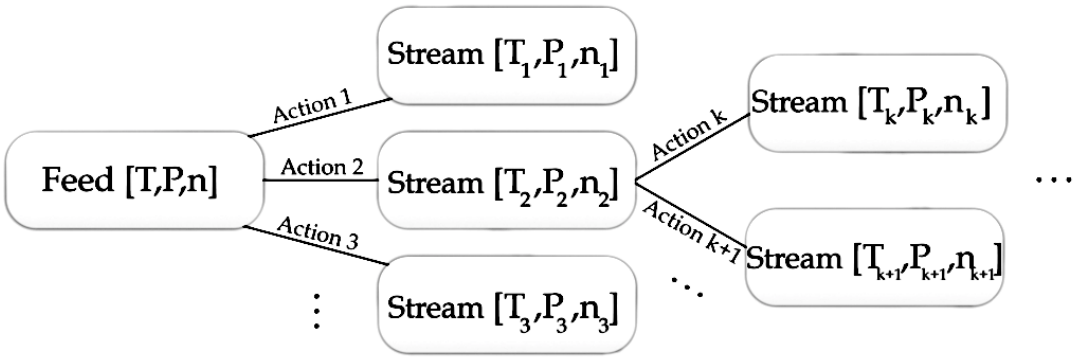

FIGURE *1: A growing decision tree in which different actions lead to distinctive thermodynamic transformations in a stream, actions are chosen decisions in the past. Desired pathways maximise profit (reward) while minimising expenses (action costs).*

The proposed framework is a learning cycle consisting of three parts : 1) an environment in which an agent explores process routes, 2) a database of past trials and their outcomes, and 3)

an agent that utilises the database to make decisions in the environment. The learning cycle is shown schematically in Figure 2. The three parts are defined as follows.

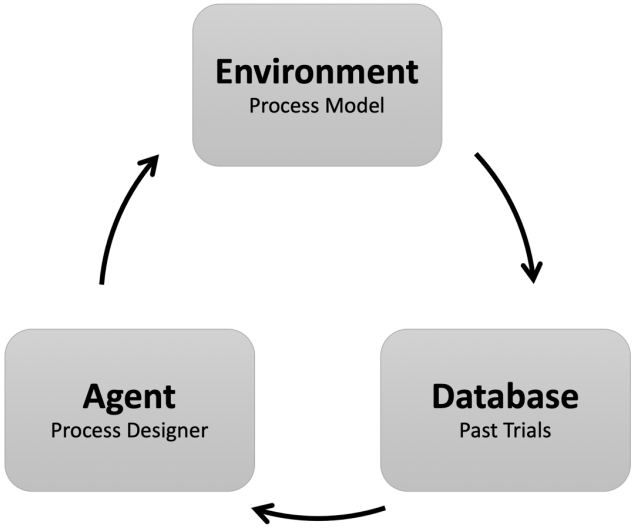

FIGURE *2: Schematic depiction of the iterative learning cycle.*

2.1.1. The Environment : Process Graph

The environment embodies the playground in which the agent can make attempts and learn new information. In this work, the environment is described by a process graph (PG). A chemical process can be illustrated as a directed graph, with nodes constituting process streams, and edges representing the chemical transformations of streams due to process operations. A process trial starts with a single feed node ; the agent takes actions that connect streams with new stream nodes. Eventually, the process forms trajectories terminating at sink nodes (indicating that a stream is sent to either products or waste).

To enable an RL agent to consider states and make decisions, the PG is framed as a Markov Decision Process (MDP), described by the set $M = \{S, A, T, R\}$, in which a state $s \in S$ is a thermodynamic property vector of a process stream, such as temperature or molar flow rates of the the species in the process. It represents the current situation with which the agent has to act. Streams change states as a response to actions $a \in A$, where A is the set of process operations and sinks available to the agent. The transition model $T : S \times A \rightarrow S$ describes the transition between states due to process operations and is governed by mass and energy balances in each

step. As such, the transition models take in the current state and action to generate the next state, satisfying the Markov property $s_{t+1} = f(s_t, a_t)$; this means the agent needs only consider the current state in deciding actions, disregarding how it got there. R is the reward (feedback) provided after completing a full trial, given here as the process profit. This feedback updates the values of the visited streams, giving :

$$V(s) = revenue - downstream\ costs \tag{1}$$

Thus, each complete trial ends with a PG containing all stream properties and values in nodes, and all action parameters and costs in edges. By definition, the optimal route is that which maximises $V(feed)$, i.e. it maximises total revenue less the processing costs. This current MDP structure will enable the agent to consider each stream as they are produced, manipulate them through actions and learn the consequences. Clearly, one can design more complex reward functions, which combine multiple objectives of the design into a single objective, thus balancing different targets e.g. economic vs. ecological, but this is out of scope for this paper.

2.1.2. The Database : Thermodynamic Graph

The structure of a database affects how information can be extracted and utilized by the agent. In this approach, the database is a record of all the visited routes, represented in the Euclidean thermodynamic state space. In other words, the database is a thermodynamic graph (TDG) containing all the previous PG trials and rewards. Linked stream nodes are from the same PG, while neighbouring stream nodes (by Euclidean distance) are similar in their thermodynamic states and need not be from the same trial, meaning the agent could have arrived to similar streams through different thermodynamic pathways, or could have diverged from similar starting streams due to taking different actions.

This representation, shown schematically in Figure 3, allows for both, simplicity in linking trial information, as well as agility in utilising similar stream data for analysis. Specifically, actions to be taken in a new stream are based on previous information about the nearest neighbors (NNs) in the TDG, which have similar thermodynamic properties like temperature and species flow rates. If the agent knows that a previous action was rewarding in a similar stream, it would be inclined to exploit that information.

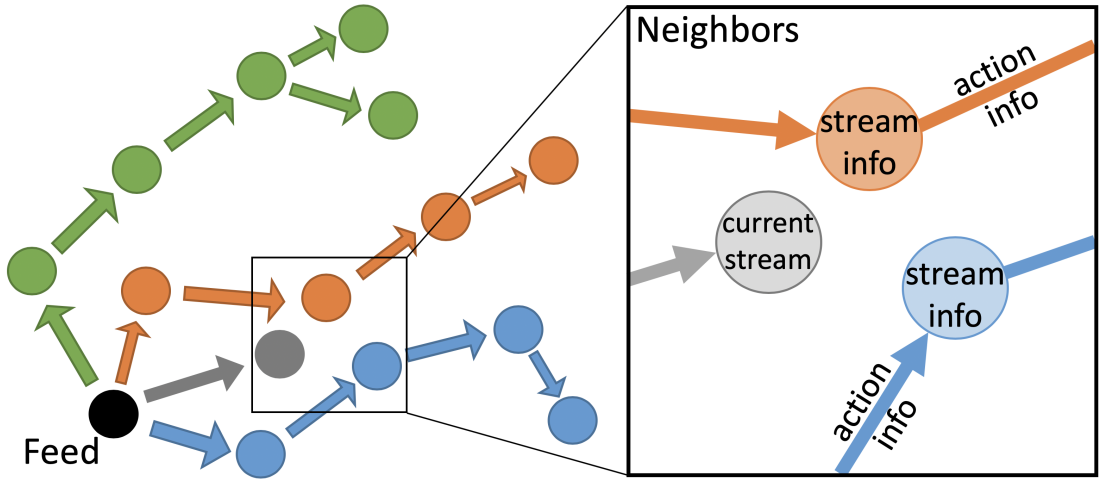

FIGURE *3: A sketch of four PG trials in TDG (left), nodes hold stream states and values, while arrows hold information on actions and costs. A decision is made on the current state of a new route, by considering valuable neighbours.*

2.1.3. The Agent

The agent is a decision maker that, given what is known about the environment, takes the best possible action. In this application, actions are chemical processing steps acted on streams. In RL approaches, the *quality* of actions is estimated using the Q-value, $Q[s, a]$, which represents the expected reward if action a was taken in state s. This Q-value is updated after each trial, iteratively converging to an optimal action policy. Different methods and approximations have been proposed to evaluate $Q[s, a]$. Further discussions on RL and its algorithmic variations can be found in [9, 10, 12].

Two types of decisions are considered in this work, discrete and continuous decisions. Discrete decisions specify the type of the process unit to which a stream is sent, or if the stream will be sent to a sink. For discrete decisions, Q-value is estimated as the ratio between the value of the best known neighbour divided by the square root of the number of NNs over which the same action was taken, see Eq. (2)

$$Q[s, a] = V_{max}(NNs)/\sqrt{size(NNs)} \tag{2}$$

This is a 'best-first' heuristic, in which the combinatorial seach is motivated by both exploiting high-value decisions (numerator) and exploring high-uncertainty decisions (denominator) [13].

It aims to balance exploration and exploitation as they represent two opposing forces in search and sampling.

Continuous decisions, on the other hand, relate to deciding unit design variables such as the reactor volume or heat duty. When considering continuous decisions, the agent finds the best neighbour, retrieves the parameters of the action taken on that neighbour, and includes a random bounded deviation to that parameter to improve exploration.

2.2. Algorithm

A pseudocode describing the three parts is shown in Algorithm 1.

Algorithm 1: A pseudocode for the learning cycle, showing a thermodynamic graph (TDG), a unit-based process graph (PG) and an agent.

```
· TDG #builds trajectory database
Initiate TDG with random trials
num_of_trials=input()
for counter in range(num_of_trials) :
    trial = PG(TDG)
    TDG ← trial #update TDG with new trial
return best_trial

· PG #starts new trial from feed node
Free_streams = [feed] #list of unprocessed streams
for stream in Free_streams :
    action, params = Agent(TDG, stream)
    if action == sink :
        Sell stream
    else :
        outlet = ODE(inlet, flux)
        Link inlet to outlet nodes with unit edges
        Free_streams ← outlet streams
Calculate trial profit and stream values
return trial

· Agent #decides action a, given stream s
for a in action_list :
    if a==sink :
        Q[s,a]=stream market value
    else :
        NNs= list_neighbors(TDG)
        Q[s,a] = V_max(NNs)/sqrt(size(NNs))
action = arg_max(Q[s,a])
params= best_NNs_parameters ± random deviation
return action, params
```

2.3. Illustrative Example

Consider the task of learning the optimal route to maximise a product B, given a reactant A and some available equipment. The task starts by initializing the TDG with random trials (initialization set), to give the agent a basis on which to make future decisions. Next, the agent would iteratively use the PG to sample actions using the knowledge it has discovered thus far. The idea is to exploit the most promising trials around which the optimal solution is more likely to be. The structure of TDG would allow for use of data by considering individual stream insformation rather than ther full trials. So, for example, an agent finding itself in the 'current stream' in Figure 3 would utilise the surrounding streams which have similar properties and would respond similarly to actions. If the agent wants to estimate how useful a reactor is, then it would retrieve data of the relevant streams on which the reactor action was chosen in the past. Let the nodes shown in the NNs in Figure 3 represent the streams that were previously sent to a reactor, these are the NNs considered when estimating Q[s, reactor] according to Eq. (2). The agent would find Q-value for every available actions, using the representative subset of the NNs for each, then choose the action with the highest Q-value. This is done for every decision over a stream. The Q-value is shaped to allow exploiting valuable actions as well as exploring unvisited ones. After a certain number of trials, the algorithm returns the best trial found in the TDG.

3. Case Study : H_2 Production

3.1. Models

Hydrogen is industrially produced from methane by means of steam reforming, water gas shift reaction and product separation. The continuous process models considered in this work are described in the following subsections.

Steam Methane Reforming (SMR). SMR converts methane (Me) and water (W) to H_2, CO and CO_2, and operates endothermically in the temperature range 900-1300 K. The process comprises of three overall reactions :

$$Me + W \leftrightarrow CO + 3H_2 \tag{3}$$

$$r_3 = \frac{k_1}{p_{H_2}^{2.5}}(p_{Me}p_W \breve{} \frac{p_{H_2}^3 p_{CO}}{K_1})/Den^2 \tag{3.1}$$

$$CO + W \leftrightarrow CO_2 + H_2 \tag{4}$$

$$r_4 = \frac{k_2}{p_{H_2}}(p_{CO}p_W \breve{} \frac{p_{H_2}p_{CO_2}}{K_2})/Den^2 \tag{4.1}$$

$$Me + 2W \leftrightarrow CO_2 + 4H_2 \tag{5}$$

$$r_5 = \frac{k_3}{p_{H_2}^{3.5}}(p_{Me}p_W^2 \breve{} \frac{p_{H_2}^4 p_{CO_2}}{K_3})/Den^2 \tag{5.1}$$

where $Den = (1+K_{CO}p_{CO}+K_{H_2}p_{H_2}+K_{CH_4}p_{CH_4}+K_{H_2O}p_{H_2O})/p_{H_2}$. Thus, the production of H_2 per mass catalyst would be $d\mathrm{n}_{H_2}/d\mathrm{m}_{cat} = 3r_1+r_2+4r_3$. Rate expressions and their parameters are described in detail in [14].

Water Gas Shift (WGS). WGS is reaction (4) taking place in SMR. It is performed on a separate catalyst to convert the side product CO and maximise H_2 production. There is a range of possible kinetic models depending on the type of catalyst used. The model used here operates at low temperatures (500-600 K). The rate expression is given by[15] :

$$CO + W \leftrightarrow CO_2 + H_2 \tag{6}$$

$$r_6 = \frac{kK_{CO}KW[P_{CO}P_W - \frac{P_{CO_2}P_{H_2}}{K_{eq}}]}{(1 + K_{CO}P_{CO} + K_W P_W + K_{CO_2}P_{CO_2})^2} \tag{6.1}$$

Heat Transfer (HX). A simplified heat exchange model was used, giving by :

$$Q = c_p \times n_{tot} \times \Delta T \tag{7}$$

Pressure Swing Adsorption (PSA). Pressure swing adsorption is a complex transient cyclic process. To reduce computational overhead, the model was substituted with a flat process giving :

- H_2-rich stream : a pure stream with 90 mol% of H_2 in the inlet.
- Waste stream : contains 10 mol% of the inlet H_2 in addition to all other components.

H_2 Separation Membrane. H_2 flux through the membrane was modelled using Sieverts' law given in the following expression [16].

$$J = const. \times e^{const./T} \times (p_{reactr} - p_{permt}) \tag{8}$$

The partial pressure of H_2 in the permeate region (p_{permt}) is neglected.

3.2. Implementation

A stream change due to actions is given to the agent by the environment's transition function s'=f(s,a), which is solved in the PG using the process models given above. Pressure was kept constant at 1 atm throughout the process. In this work, only extreme NNs sizes were found to affect results, so neighbors were set to be those nodes inside a hypercube whose center is the desired stream and whose sides span 8% of the possible range for each property (e.g. $N_{Me} \in [0 - 500] mol/s$, so the limit for a neighbour is $\pm 20 mol/s$ from the state of interest). The number of NNs should be large enough to include informative nodes, but small enough to exclude irrelevant nodes.

Two approaches to process design are considered in this work : (1) actions based on conventional unit operations, and (2) actions based on combined phenomena which produce a single hypothetical multifunctional unit. In both environments, trials start with the feed stream at state s= [T=900 K, N_{Me}= 500 $mols^{-1}$, N_W= 2,000 $mols^{-1}$, N_{H2}= 500 $mols-1$, N_{CO}= 0 $mols^{-1}$, N_{CO_2}= 0 $mols^{-1}$]. Although there is an overlap, the two approaches attempt to solve different problems :

1. Unit Operations : In this approach, the agent is not given a superstructure a priori, but is asked to design a profitable process given the feed and the available actions only. Actions are decided via discrete decisions for unit types, and continuous decisions for unit design variables, as described in 'The Agent' section. The set of available actions and their parameters are given as A :{'SMR' :[cat. mass], 'WGS' :[cat. mass], 'HX' :[q], 'HXSMR' :[cat. mass, q], 'HXWGS' :[cat. mass, q], 'PSA', 'sink'}. A max limit of 6 units was imposed per trial ; if a trial reaches 6 units, all free streams are sent to sinks and the trial terminates.

2. Phenomena : Here, an intensified multi-functional unit is defined a priori, and the agent is asked to impose fundamental phenomena by making four continuous decisions for the design variables : heat flux, mass of the SMR catalyst, mass of WGS catalyst, and surface area of the membrane, A = [HX, SMR, WGS, J]. The intensified unit is divided into multiple sections, where each section can have different values for the four parameters, e.g. the agent could choose large heat influx at earler sections of the reactor, and low heat at later sections. A four-section

unit is shown in Figure 4 as an example. For each trial, the agent makes a decision for each of the four parameters, in each of the sections, to optimise for total profit.

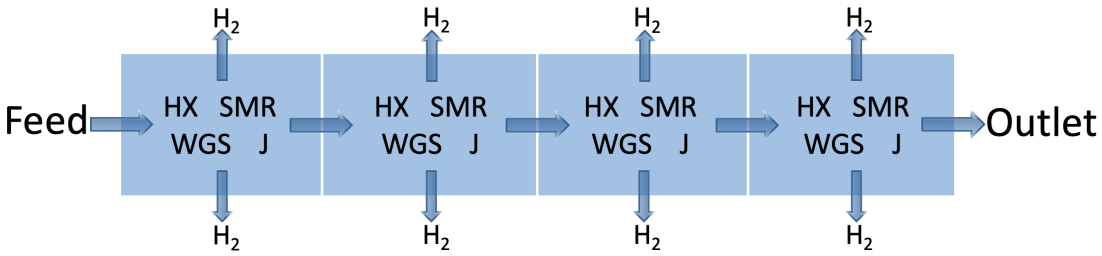

FIGURE *4: An illustration of a hypothetical multi-functional unit, having four sections, with four possible phenomena in each.*

3.3. Problem Statement

The optimisation problem is formulated as

$$Obj = \max_{a} \ (revenue - costs) \tag{OP}$$

s.t.

Energy and component balances per unit/section.
Constitutive equations : Eqs. (3)-(8).
Bounds per unit/section :

$$\begin{aligned} -50 \leq \text{HX duty} \leq 50 \text{ kJ} \\ 0 \leq \text{SMR cat.} \leq 500 \text{ kg} \\ 0 \leq \text{WGS cat.} \leq 500 \text{ kg} \\ 0 \leq \text{membrane area} \leq 10 \text{ m}^2 \end{aligned}$$

Revenue from selling a stream was only considered if it had pure H_2, otherwise it was sent to waste and would have a value of zero. Processing costs were simplified to be linear functions of materials/heat prices for parametrized actions (e.g. cost(HX) [$] = heat pricing [$/kJ]× heat duty [kJ]), and a constant cost for PSA. This pricing scheme, albeit simple, produces a nonlinear profit space due to the nonlinearity of the process models.

Certain process constraints were also imposed : the outlets of a reformer, a WGS reactor and heat exchanger do not exceed 1500, 1100 and 1900 K, respectively. No outlet should be below 400 K either. Rather than forcing the agent to only operate under these constraints, it is allowed to explore the entire thermodynamic state space, but would receive a large penalty for violated constraints, and end the trial. The agent was only bound in continuous variable decisions.

4. Results and Discussion

4.1. Optimisation Based on Unit Operations

Four search heuristics were tested when implementing the agent in a unit-based environment :

1. choose the best discrete action using Q-value, and include a 25% deviation to the best continuous variable,
2. choose the best discrete action, add a 10% deviation to the best continuous variable,
3. choose the best action, and randomly decide continuous variable, and
4. choose random actions and variables.

First, the agent builds a basis for the TDG by running 400 random trials (initiation set). Then it proceeds to decide actions using the search heuristics over the available trial data. Heuristics were tested over 100, 300 and 800 additional trials (a sampling set), results for the sampling set are shown in Figure 5. Over the different sizes, the same trend is seen : a 25% deviation yields the best process profit, likely due to its ability to balance exploration and exploitation in the search for suitable parameters.

To give a practical comparison with the results, an optimal profit of 29.2 \$ s^{-1} was found manually using trial-and-error and the authors' own knowledge of the process. This required roughly two minutes – about the same time taken by the algorithm to run 300 trials. If, due to some error, a trial simulation did not converge, then the trial information was discarded, and the agent would start the next trial. For example, if the agent was asked to produce 100 trials but one of them crashed, then it will discard that failed trial and return the best results out of the 99 successful trials.

Over the different trials, different optimal processes were reported. This is due to the stochastic nature of the search for parameters, which affects the best-first approach to deciding appropriate units as well. Additionally, some unit choices were made purely based on their arbitrary presence in profitable random trials in the TDG, even if those units had negative effect on the overall profit. This is due to the naïve nature of decision-making using the best neighbouring pathway, which does not consider the utility of a particular action or its effect on the overall process.

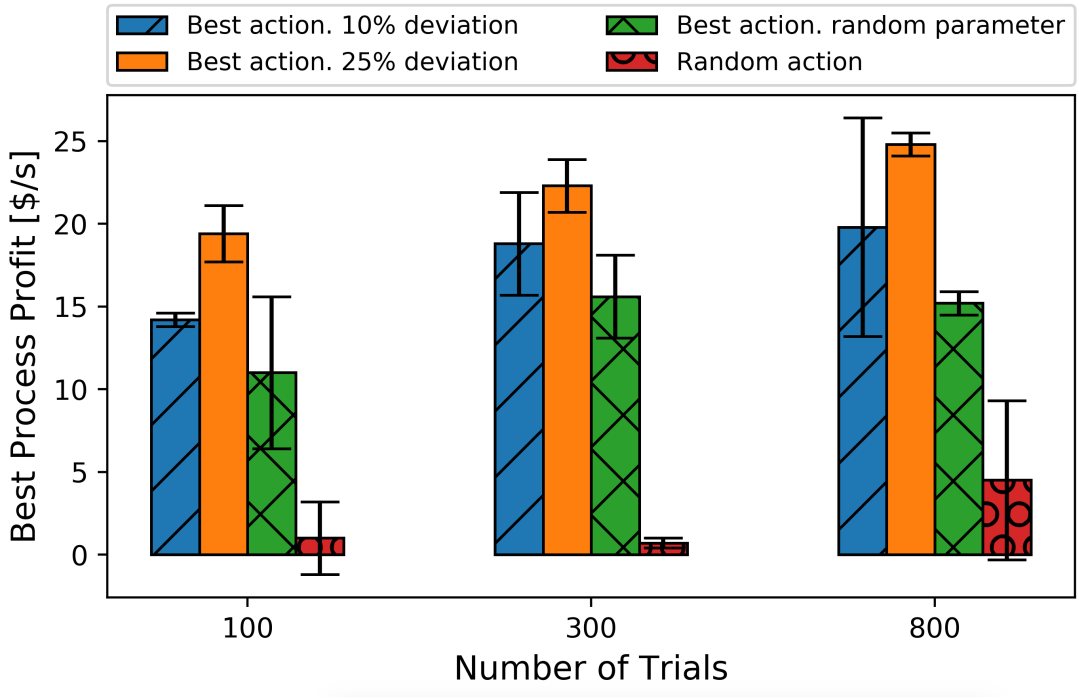

FIGURE *5: Average best process profits over 100, 300 and 800 unit operation trials with 10, 5, and 2 re-runs, respectively.*

4.2. Optimisation Based on Phenomena

Preliminary runs confirmed the results of the unit-based approach, so 25% deviation was set as the search heuristic for this approach. Next, the agent's performance was compared with a standard optimisation package (Scipy) in Python. Two gradient methods were used for benchmarking : sequential quadratic programming (SLSQP) and a nonlinear quasi-Newton approach (L-BFGS). Performance was assessed in two situations : a small sampling size, and a large search space.

First, the agent was stopped after 100 trials in multifunctional unit with 2, 4, 6 and 8 sections. The results in Figure 6 show a decreased profit as the number of unit sections increase, as expected in an increasingly complex problem. The agent maintained computational speed while still discovering profitable solutions ; random search was never able to find profitable solutions in the 8-seciton unit. The optimisation algorithms found the best solutions, L-BFGS maintained performance, while SLSQP maintained computational time.

Notably, the gradient methods required a suitable initial guess as random guesses mostly

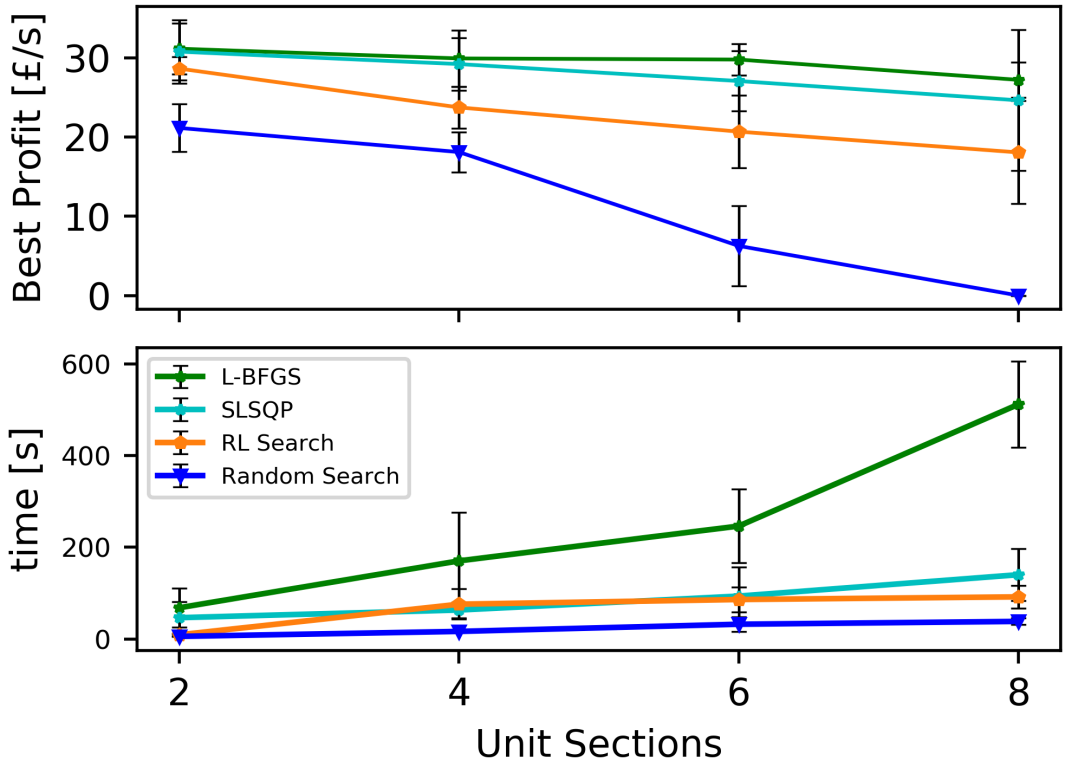

FIGURE *6: Best profit over 100 trials for RL and random search, compared with two gradient methods. RL search scaled well while random search could not discover any profitable 8-section solutions. Results repeated 10 times for RL and random searches, 5 times for optimisation packages. 100 trials were used for the initiation set.*

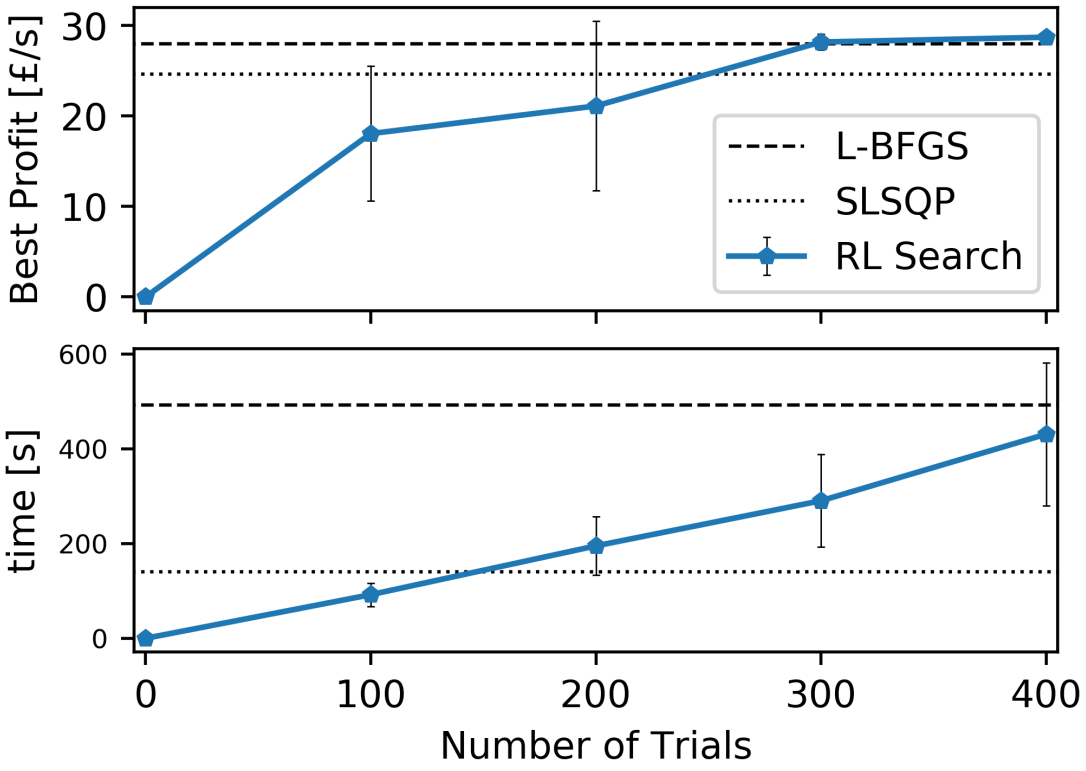

FIGURE 7: *RL search profit and time as the number of trials increase for an 8-section multifunctional unit. Gradient method benchmarks are plotted. Increasing the number of trials increases both the initiation set and the test set for the agent, results were repeated 20 times.*

led to poor performance. So initial guesses for each flux were manually set to be within small bounds to ensure a stable performance. For example, the gradient methods were constrained to initial SMR guesses between $0 - 50\, kg_{cat}/section$, while the stochastic search allowed the agent to robustly find profitable solutions within a guessing range of $0 - 500\, kg_{cat}/section$.

Second, to test its ability to find solutions in a combinatorically large space, the agent was allowed different trial sizes for the 8-section unit. Results in Figure 7 show the agent repeatedly outperformed the optimisation algorithms after 300 trials, within comparable computational time. The agent, by exploiting TDG data and exploring encouraging regions, could frequently discover optimal pathways.

In terms of the resulting process designs, the main difference between the agent and the gradient methods is that the latter produces exact (local or global) optima, while the best solutions by the agent are not necessarily local optima. For this reason we find that it naively includes

detrimental actions only because those actions were arbitrarily present in profitable pathways, e.g. having WGS catalyst in conditions where the reaction is not favoured. Learning the environment model would circumvent this issue by only considering actions that are expected to improve a stream. It is also concevable to exploit synergy between RL and gradient methods, e.g. improve the agent'a learning with local optima from gradient methods, or improve gradient methods by using the agent's output as an initial guess.

The reported approach could provide informative data to build models for the environment. Figure 8 demonstrates the agent's attentive reduction of methane flow through a 4-section unit, compared to random search. Given 100 random trials as an initiation set and sampling set of 100 trials, the RL agent focuses its sampling (with up to 25% deviation) on flux combinations that are known to deplete the reactant and lead to low methane flow at the outlet. Random search results are scattered over the design space, as it was not guided by an objective value (profit in this case). Focusing the sampling on regions of interest would facilitate building data-driven models, which were shown to accurately learn complex process models [1, 8].

A few key areas for future work are worth mentioning. Further development will incorporate rigorous process models e.g. VLE calculations and support intra-/inter-unit recycling of mass and heat (described in [17]). Also, while profitable pathways were frequented in this work, the agent was rather simplistic in learning action values that guide sampling. More complex problems would require a far more efficient search. Future work will investigate advanced concepts in structural pattern recognition and data-driven models to exploit flux interactions and strategic processing routes, instead of following pre-defined search heuristics and value estimators. Additionally, combining features of unit operations and phenomena approaches would give a multi-level design space. Furthermore, a multi-objective formulation of the problem could enable optimising both process-specific (economic, sustainability) and search-specific (model learning and sampling strategy) targets.

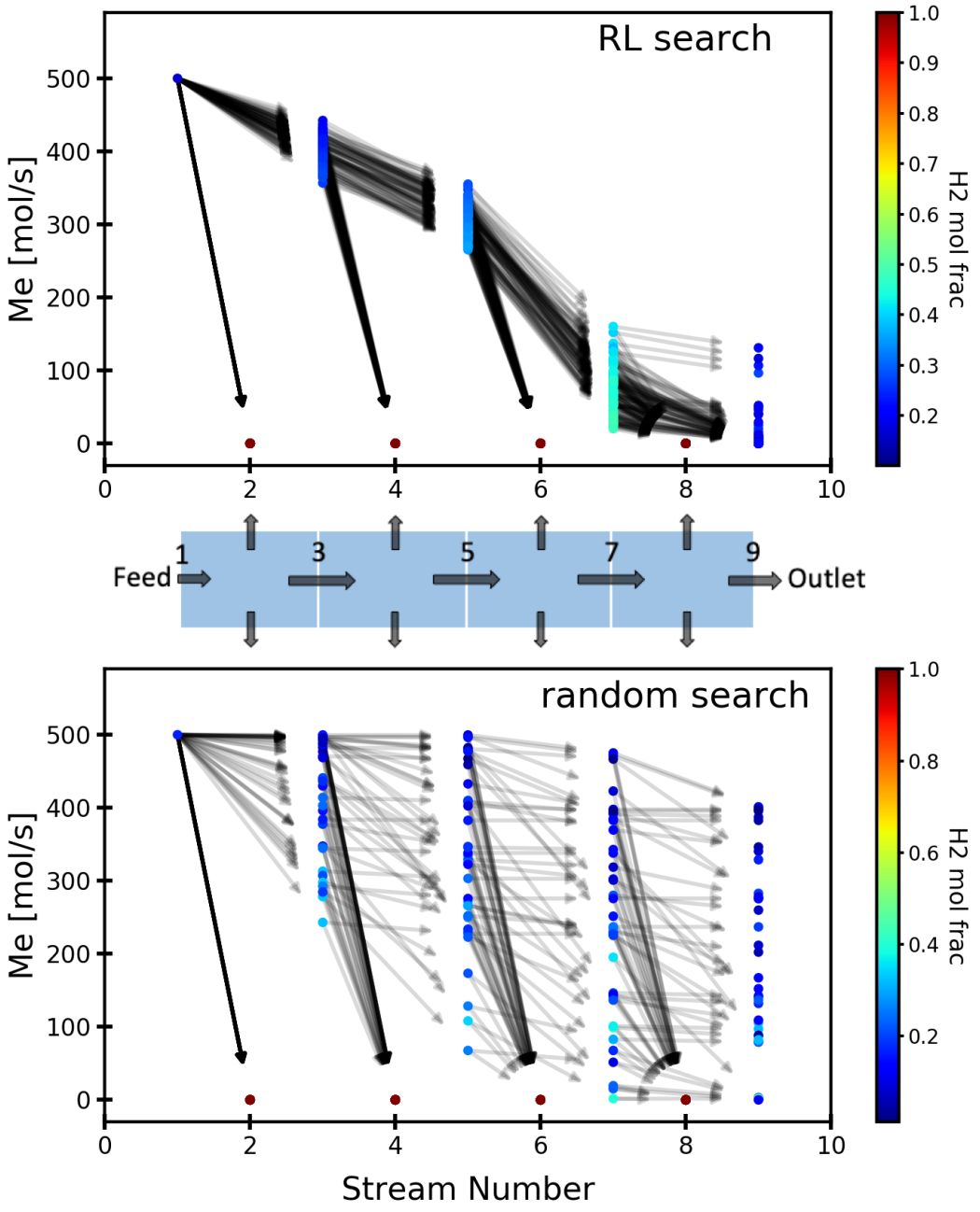

FIGURE 8: *Change in methane as it flows through a 4-section intensive unit over a test set of 100 trials. Each trial starts with 500 mol/s methane. Arrows connect the stream states in a trial, e.g. stream 1 is sent to the first section, producing streams 2 and 3 which contain the separated hydrogen and the feed to the next section, respectively.*

5. Conclusions

The outlined framework employs concepts from reinforcement learning to search for optimal process designs. A sampled process route is depicted in a graph of the thermodynamic state space, representing the transformation of stream nodes due to unit operations or fundamental process phenomena. The learning agent iteratively improved profit through trial-and-error, stochastically narrowing its search to the most promising trajectories in the state space. A rudimental implementation of the agent was found superior in both performance and robustness compared to standard gradient methods in a hydrogen production case study, inspiring further improvement using sampling strategies and pattern recognition techniques. Potential applications include quickly finding solutions in simple problems, and finding novel solutions in more complex problems.

Acknowledgements

We thank Professor Pietro Lio for the valuable discussions on this topic. AK is grateful to Saudi Aramco for his PhD scholarship.

Références

[1] G. Stephanopoulos, C. Han, Intelligent systems in process engineering : a review, Comput. Chem. Eng. 20 (6/7) (1996) 743–791.

[2] H. Freund, K. Sundmacher, Towards a methodology for the systematic analysis and design of efficient chemical processes, Chemical Engineering and Processing 47 (12) (2008) 2051–2060.

[3] S. Demirel, J. Li, M. Hasan, Systematic process intensification using building blocks, Comput. Chem. Eng. 105 (2017) 2–38.

[4] P. Lutze, R. Gani, J. Woodley, Process intensification : a perspective on process synthesis, Chemical Engineering and Processing 49 (6) (2010) 547–558.

[5] Q. Chen, I. Grossmann, Recent developments and challenges in optimization-based process synthesis, Annu. Rev. Chem. Biomol. Eng. 8 (1) (2017) 249–283.

[6] L. Cheng, E. Subrahmanian, A.W. Westerberg, Design and planning under uncertainty : issues on problem formulation and solution, Comput. Chem. Eng. 27 (6) (2003) 781–801.

[7] A. Westerberg, A retrospective on design and process synthesis, Comput. Chem. Eng. 28 (8) (2004) 447–458.

[8] J. Cagan, I. Grossmann, J. Hooker, A conceptual framework for combining artificial intelligence and optimization in engineering design, Res. Eng. Des. 9 (1) (1997) 20–34.

[9] S. Russel, P. Norvig, Artificial intelligence : a modern approach, 3rd Edition, Prentice Hall, New Jersey, 2010.

[10] R. Sutton, A. Barto, Reinforcement learning : an introduction, The MIT Press, Cambridge, MA, 1998.

[11] V. Venkatasubramanian, The promise of artificial intelligence in chemical engineering : is it here, finally ?, AIChE 65 (2) (2019) 468–475.

[12] L. Kaelbling, M. Littman, A. Moore, Reinforcement learning : a survey, J of Artif. Intel. Res. 4 (1996) 237–285.

[13] C. Brown, E. Polwey, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of monte carlo tree search, IEEE Transactions on Comp. Intel. and AI in Games 4 (1) (2012) 1–43.

[14] J. Xu, G.F. Froment, Methane steam reforming, methanation and water-gas shift, i. intrensic kinetics, AIChE 35 (1) (1989) 88–96.

[15] A. Criscuoli, A. Basile, E. Drioli, An analysis of the performance of membrane reactors for the water gas shift reaction using feed mixtures, Catalysis Today 56 (2000) 53–64.

[16] AS. Kyriakides, D. Ipsakis, S. Voutetakis, S. Papadopoulou, P. Seferlis, Modeling and simulation of a membrane reactor for the low temperature methane steam reforming, Chem. Eng. Transactions 35 (2013) 109–114.

[17] M. Xie, H. Freund, Fast synthesis of optimal chemical reactor networks based on a universal system representation, Chemical Engineering and Processing : Process Intensification 123 (2018) 280–290.

Abbreviations

CO : carbon monoxide.

CO_2 : carbon dioxide.

c_p : heat capacity [kJ/mol].

H_2 : hydrogen.

HX : heat transfer phenomena.

J : flux through membrane [mol/m^2s].

k_i : reaction i rate coefficient.

K_i : equilibrium constant for reaction i.

K_j : adsorption constant for component j.

Me : methane.

NNs : nearest neighbours.

p_i : partial pressure of component i in the rector [atm].

p_{reactr} : H_2 partial pressure in reactor [atm].

p_{permt} : H_2 partial pressure in sweep gas [atm].

PG : process graph.

PSA : pressure swing adsorption.

Q : heat flow [kJ/s].

r_i : rate of reaction i [mol/g_{cat}].

RL : reinforcement learning.

SMR : steam methane reforming.

T : termperature [K].

TDG : thermodynamic graph.

W : water.

WGS : water gas shift.